Appears in the proceedings of SPIE ‘96, Stereoscopic Displays and Virtual Reality Systems

VR + AI = intelligent environments:
A synergistic approach to engineering design support

Rudolph P. Darken
Naval Research Laboratory
Washington, D.C.
darken@enews.nrl.navy.mil

Christian J. Darken
Siemens Corporate Research
Princeton, N.J.

ABSTRACT

Both VR and AI have the potential to be huge productivity enhancers for engineering design, and in complementary ways. VR is a visualization tool allowing users to comprehend complex spatial relationships among many variables. AI is an exploration tool capable of finding and exploiting relationships which are very difficult to visualize, but is most effective with few variables. Using engineering design as an example, we explore how VR and AI might be integrated to yield productivity gains greater than either might alone.

The typical engineering design cycle for a complex system involves multiple passes through design, simulation, and analysis phases. VR is used to visualize a design simulation while AI is used to assist in the subsequent redesign. The role of the VR subsystem is twofold; it visualizes the data for analysis and problem diagnosis such that it is easily comprehended by the engineer, and it provides a mechanism by which the engineer can describe how the design is to be improved in the next iteration. The AI subsystem then acts on the redesign descriptions to suggest design modifications. These suggestions are integrated with direct modifications from the user, and the redesigned system is simulated again. The synthesis between the VR and AI subsystems results in a closed loop design system capable of effectively undertaking complex engineering design tasks.

Keywords: virtual reality, artificial intelligence, engineering design

1. INTRODUCTION

For all of its successes, the last two decades of artificial intelligence (AI) research have revealed significant limitations in AI techniques which are still only beginning to be overcome. Two reasons why AI techniques often fail where humans excel are easy to state: the power of the human visual system and the wide-reaching human knowledge-base. Seen from the point of view of AI, virtual reality (VR) seems almost tailor-made to overcome these limitations by providing the eyes of a human expert the ability to interact with AI modules in a natural way. Similarly, while VR research has made great strides in the past several years, it has largely been used as a visualization tool or generic simulation system. Seen from the point of view of VR, the addition of AI in engineering design applications would greatly increase the efficiency of the engineer by significantly decreasing the length of the design cycle. By using VR and AI together in a closed-loop system, we suggest that design engineers can focus their efforts on candidate designs that are ‘closer to the solution’.

Notwithstanding their limitations, AI techniques are of outstanding relevance and usefulness for system modeling and optimization. One set of techniques of considerable importance is the use of neural networks and other semiparametric functions for general purpose modeling. It is worth pointing out, for those not already aware, that the ability of such tools to represent almost any system (by representing its time evolution functions, i.e. differential equations or discrete update formulae) has been mathematically proven (see [10], for example). What is not always realized, however, is that for high-dimensional systems the computational and data requirements of such techniques tend to make them infeasible. Unfortunately, the collapse of AI algorithms as dimensionality increases is the rule rather than an exception. Therefore the potential use of the human eye to interactively assimilate high dimensional data and direct the attention of an AI system to limited subproblems is an exciting prospect.

A well-studied, but difficult to characterize, problem in AI is the issue of making expert knowledge accessible to an AI system. The usual approach is to try to “bottle” this knowledge and store it up for use by the system as needed. While such approaches exist, they are often excessively costly, requiring the design of an appropriate knowledge representation and significant commitment by both an expert and a knowledge engineer. Furthermore the resulting system performance often leaves much to be desired. Active supervision by an expert is clearly to be preferred from a performance point of view, and may also be less consumptive of resources in many task environments.

VR has proven its worth in visualization, simulation, and training, but has fallen short of supplying the type of assistance that AI can provide. What we find are examples of VR as a tool to assist designers and engineers in analyzing designs or data sets but the assistance is one-way. VR is used as an interactive visualization system that does not directly affect the design at all but rather allows only interaction with the visualization component (See figure 1). The user identifies problems and evaluates the design using VR to visualize simulated data. The diagnosis and subsequent alterations to the design are then done manually, often based on intuition and experience. Typically, the designer iterates back and forth between the VR visualization system and design tools that help construct redesigns based on problems identified by the visualization and the designer's "best guess" as to their cause.

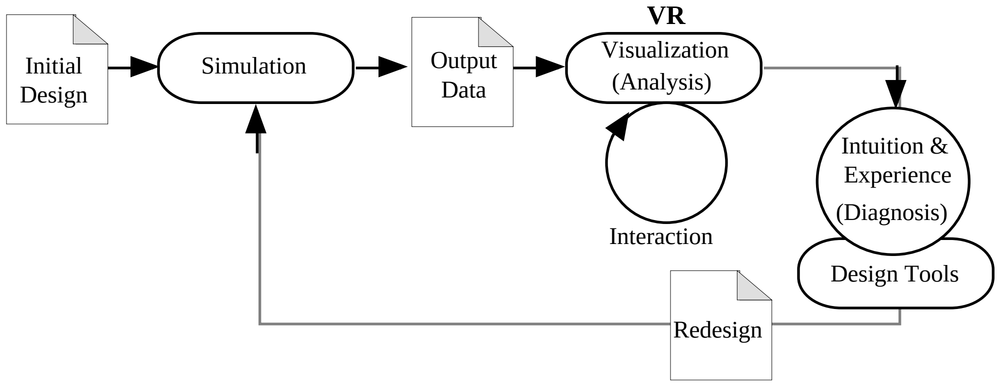

Figure 1: A typical system configuration using VR as an analysis tool only. The system is not a closed loop as the designer iterates between the analysis and diagnosis phases, both of which are performed manually.

What we want is a system that closes this loop, relying less on intuition to construct redesigns and more on a systematic investigation of candidate designs likely to solve the design problem. By using AI to focus the efforts of the user on better designs, convergence on a final design will occur more quickly.

2. CHARACTERISTICS OF A SUITABLE PROBLEM DOMAIN

It is certainly true that not all problem domains require the use of VR and AI to produce a solution. Many applications do not possess the complexity to require AI, or the spatial characteristics to require VR, or possibly, they would not have the necessary payoff to make it worthwhile. However, we believe that there is a characteristic set of problem domains that would yield great improvements in cost, time, and proficiency.

For any given design, we assume there exists a set of criteria by which its goodness can be objectively evaluated. It is often through visualization, mediated by VR, that these evaluation criteria are observed (See figure 2).

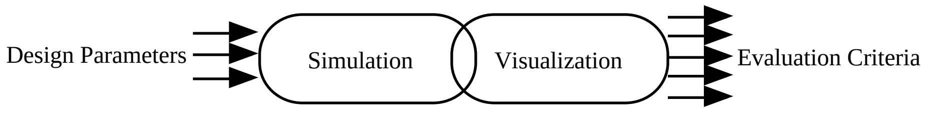

Figure 2: The non-intuitive relationship between the design parameters and the evaluation criteria is observed via a visualization of simulated data.

The primary requirement of a suitable problem domain is that there exists a non-intuitive cause/effect relationship between the input design parameters and the evaluation criteria. For example, the design of an automobile body might be described in terms of a detailed CAD model (i.e. geometric data). That model might be visualized in a virtual wind-tunnel such as that described by Bryson and Levit [3] to determine points or regions exhibiting excessive drag or wind resistance which are, in part, the model's evaluation criteria. It is difficult, if not impossible even for an expert designer, to know specifically what the effect will be of a geometric alteration on the resultant aerodynamic measurements. Thus there is a need of assistance which the right AI algorithms could possibly provide.

A secondary, but also necessary, requirement is that the problem domain exhibit spatial attributes necessitating the use of VR as a visualization tool. If this is not the case, then a suitable solution can probably be found without the added expense of VR by using a traditional desktop system. VR is most effective on highly dimensional data that is typically difficult to comprehend in a flat, two-dimensional form. There are many design tasks in this category, in particular the design of systems which manipulate a spatial field (e.g. electromagnetic, fluid flow, distribution of a substance like a bacterium or chemical or of a quality like temperature). While not strictly "engineering design", the development of military strategies (e.g. electronic warfare tactics) also fits this paradigm, as we will describe in detail later.

3. SYSTEM DESCRIPTION

The design cycle for a complex system typically relies upon the repeated use of a simulator (see figure 3). An initial design, followed by successive redesigns, is simulated to give the user insight into its strengths and weaknesses. The initial design is described in a specific language for the given domain. That is, an automobile body would be described in terms of a CAD or other geometric data language while a military engagement would be described in terms of forces, flight paths, event times, or other parameters needed for that application. The language is defined by the application domain. We then propose to use VR to visualize the results of each simulation and to provide a convenient mechanism for the user to describe what needs to be changed. An AI subsystem then acts on the description to suggest design modifications. These suggestions are integrated with direct modifications from the user, and the redesigned system is simulated again.

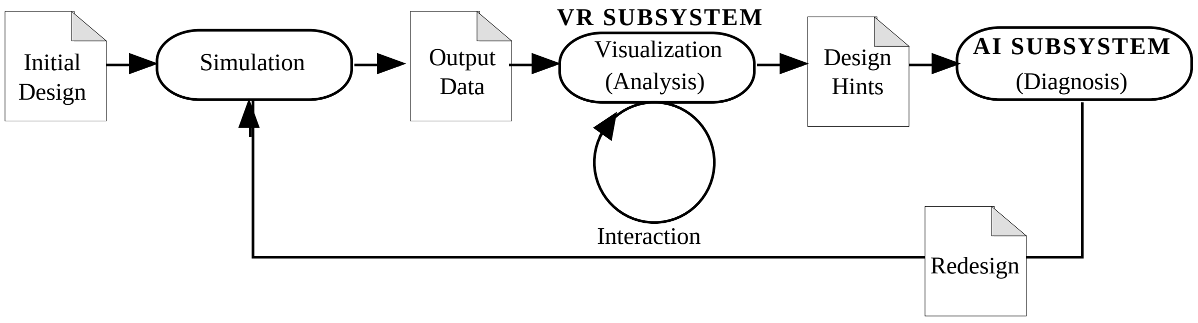

Figure 3: The proposed system configuration with the VR subsystem used for analysis and the AI subsystem used for diagnosis. This system is a closed loop as the designer iterates between the analysis and diagnosis phases; only the analysis phase is performed manually. Redesigns can be constructed automatically from the AI subsystem based on user suggestions in the analysis phase.

3.1. The VR Subsystem

The VR subsystem consists of the simulation and visualization components of the overall system. Its objectives are first, to present the simulated data for analysis by the user, and second, to allow a mechanism by which the user can make corrections and/or suggestions to the AI subsystem so that the next generation of candidate solutions can be computed.

The simulation phase is necessary because it is the tie that binds the design inputs to their corresponding outputs for evaluation. The simulation for a complex system would probably consist of a number of non-intuitive relationships between variables, possibly including partial differential equations or the like. The resultant data is then visualized for presentation to the user. The simulation and visualization phases might be executed in parallel provided that the simulation is not too computationally demanding. However, this is not a requirement. Many of the problem areas that would most benefit from the synergy of AI and VR are high dimensional and computation intensive. In these cases, the simulation can be executed asynchronous to the visualization phase which must be run in real-time. It is the visualization phase where VR has really proven its worth.

We will not dwell on the merits of VR as a visualization tool but will only remark on the advances in recent years in visualization techniques for virtual environments. Probably the most notable example of a VR visualization tool is the virtual windtunnel [3]. The virtual windtunnel showed how an immersive environment could be used to explore abstract data in ways that would be otherwise impossible. Other examples include those involving molecular modeling [1, 4], architectural walkthroughs [2], and astronomic data, fractals, regional weather and others [5].

In addition to helping the user identify problem areas in the design, the VR subsystem must facilitate the communication of corrections and suggestions to the AI subsystem. Notice that this functionality is somewhat different from interaction with the visualization component. The user will certainly want to interact with the visualization system via queries (e.g. “Highlight all the regions where chemical A has concentration greater than X and the temperature is less than Y.”) or changes in visualization technique or parameters (e.g. “Change the resolution of the mesh to 1 cubic centimeter.”) to obtain all the information available in a form most suitable to its use. Although this type if interaction is important to the evaluation of a design, what we are concerned with here is the user’s input to the AI subsystem.

The most obvious form of interaction with the AI subsystem is direct corrections to the design input parameters. The user may wish to change the shape of a curve on an automobile body by sweeping it out with a rough curve. A less obvious, but equally important, form of interaction takes the form of suggestions or guidance for the pending AI search. Suggestions can be made, such as “I think the problem is in this region” (sweep out a region with a pointing device) or “The temperature at this point (specify a particular point in space) is too low” by interacting with the environment itself. The user might also wish to set a prioritized list of input parameters for the AI subsystem to search. For example, the user may believe the temperature variable to be more likely the cause of a problem than the pressure variable.

Actually, by specifying this information, the user is constructing an optimization function for the AI subsystem to use in its evaluation of the search space of possible solutions. This is the function that the AI subsystem uses to objectively evaluate redesigns. We will discuss this in detail in the next section.

In summary, the VR subsystem is required to be both the means for analysis of a design and the method for inputs to the AI subsystem. The user must be able to specify:

- a prioritized variable list
- modifications of the objective function
- new initial conditions to explore (change of design input variables directly) and
- regions where the solution is poor.

3.2. The AI Subsystem

The AI subsystem is responsible for interpreting hints by the user and for suggesting design revisions. The hints can be presented in some formal language (the use of natural language is not necessary). The hints contain two types of information: what problems exist with the design (e.g. “the concentration of A is too high in this region”) and how this may be corrected (e.g. “try adjusting the flow rate at the lower nozzle”). The subsystem is responsible for suggesting design revisions (e.g. via inverse modeling or direct optimization as in neural network control), in addition to those that may have been suggested by the user via the VR subsystem, which are then used as inputs into the simulator and VR subsystem to complete the closed loop system. For systems in which the simulation is fast enough, the AI subsystem may make direct use of the simulator in generating its suggestions, thus lessening its reliance on the human operator to guide the search intelligently.

Design is the art of trade-offs. The usual case is that two or more subsets of the design criteria oppose one another. At the very least, the constraint that cost must be kept low usually opposes performance criteria. We may consider that design is done in a “design space” where the dimensions of this space include the parameters of the design which can be adjusted (e.g. the thickness of a construction material) and the design criteria (e.g. cost). Every conceivable design is a point in this space, and the set of all designs is usually a surface in this space. The task of AI is to help steer the user from an initial design towards regions of the space that better meet the design requirements. This can be done either directly, by learning the performance surface, or by more subtle means.

AI is a collection of tools which must be customized to the task at hand. The most useful tools for interpreting user hints come from research on formal and natural languages. Given that most engineering design scenarios are restricted from a linguistic point of view (thus limiting terminological ambiguity), the existing language techniques might be expected to perform fairly well. Graphical data, which we expect would mostly be in the form of curves swept out by the user, can be represented as functional coefficients (e.g., neural network weights) for further processing by the AI subsystem.

For design revisions, the relevant tools are those for system modeling and optimization. Optimization techniques (particularly numerical ones) are not always considered a part of AI, but it is reasonable to do so, especially since these techniques have great currency in modern AI research, which has also involved itself in extending and customizing the techniques for AI applications. When the domain is discrete, the whole range of AI search techniques may be called upon to perform optimization. The available tools for modeling are such semiparametric function representation schemes such as neural networks of various flavors, and others borrowed from statistics and engineering such as MARS, splines, wavelets, and even, despite some significant limitations, the traditional ones such as polynomials and Fourier transforms.

Since the AI toolkit is in fact so large, we focus on applications of a subset, neural networks and related techniques, to the design engineering task. Neural networks are a function representation and learning technique. Other function learning techniques, in particular methods for learning piece-wise linear functions, belong in the same class as neural networks, and all have possibilities for application in this area. There are several distinct ways in which function learning is applicable to the task of design engineering. The most obvious is simply to represent a function which itself needs to be optimized. We present such an example in the sample task below. Another use is to learn the feasible design surface in design space,, i.e. the design criteria as a function of the design specifications. The example cited in [9] is that of learning the output power and fuel consumption of engines as a function of such parameters as the engine speed, displacement, compression, and timing. A third use is relevant when the system to be evaluated has a time evolution which can be represented by ordinary differential equations. Then neural networks can provide a differentiable model for the process dynamics, allowing the parameters to be optimized to be tuned by the large and well-developed family of gradient-based methods [8].

4. A SAMPLE PROBLEM DOMAIN: Force-on-Force Naval Mission Planning

The example problem we will present here is that of a Naval mission planning tool used to plan a strategic air strike. In this case, the mission is a relatively simple air attack of a land-based target. The objective (in the planning phase, not the execution phase) is to construct a plan with the highest probability of successful elimination of the target with the lowest probability of loss of any portion of the strike force. We have purposely selected an example that is simple enough to describe in some detail here understanding that an actual mission planning tool would be far more complex, thus making this problem domain even more suitable to this technology than our example might show. We comment on the scalability of this problem in the next section.

4.1. Problem Description

Before beginning, the user is given an initial set of constraints. These may be logistical or practical problems to be considered as constants in the design process. For this example, we assume the following constraints are given:

- The origin of the strike force
- The location of the target
- The final destination of the strike force
- The location of any known land-based hostile missile sites[†]
- The spatial distribution giving probabilities for mobile missile sites in the region

The user of this system interactively describes a mission plan (i.e. the initial design) in terms of:

- The locations of two jamming aircraft (we take these to be point locations)
- The strike trajectory of the strike force (a single aircraft)

The solution should minimize the probability of detection of the strike force while also minimizing the exposure of the jamming aircraft to possible threats (defined by the probability distribution of mobile missile sites along with the known stationary missile sites). As mentioned earlier in reference to general design problems, these two criteria directly oppose each other. The strike might have a high probability of success if we were to move the jammers in very close, opening up a wide corridor for the flight path (See figure 4). This, however, would place the jamming aircraft in significant jeopardy from mobile missile sites. However, if the jammers are moved too far out, the air strike will likely fail because the flight corridor is not properly opened (See figure 5) What we really want is an optimal balance between these two criteria.

The design parameters are not intuitively related to the performance criteria. That is, the user does not know specifically how movement of the jamming aircraft will affect the detection ranges of the missile sites. Accordingly, the user also does not know what the best trajectory would be because the detection ranges for each site change as the position of the jammers changes. The problem is further complicated by the fact that the probability distribution of mobile sites

[†]. We assume all missile sites are also radar sites.

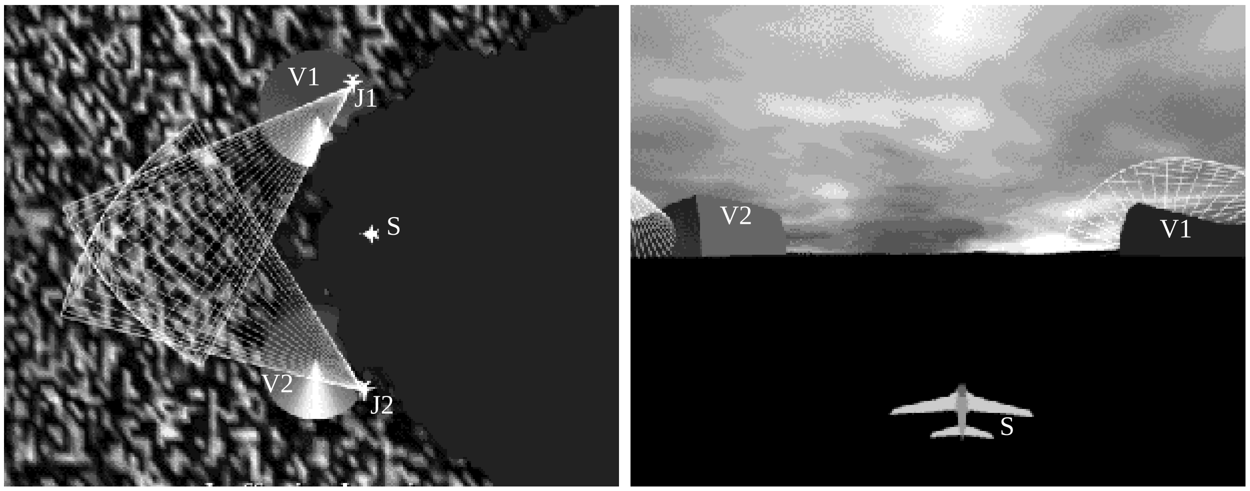

Figure 4: [Left] An example design configuration with the jammers (J1 and J2) too close to the SAM sites. Although a large corridor is opened for the strike aircraft (S) between the detection volumes (V1 and V2), the jamming aircraft are themselves detected and thus are in jeopardy. [Right] The immersive view of this configuration.

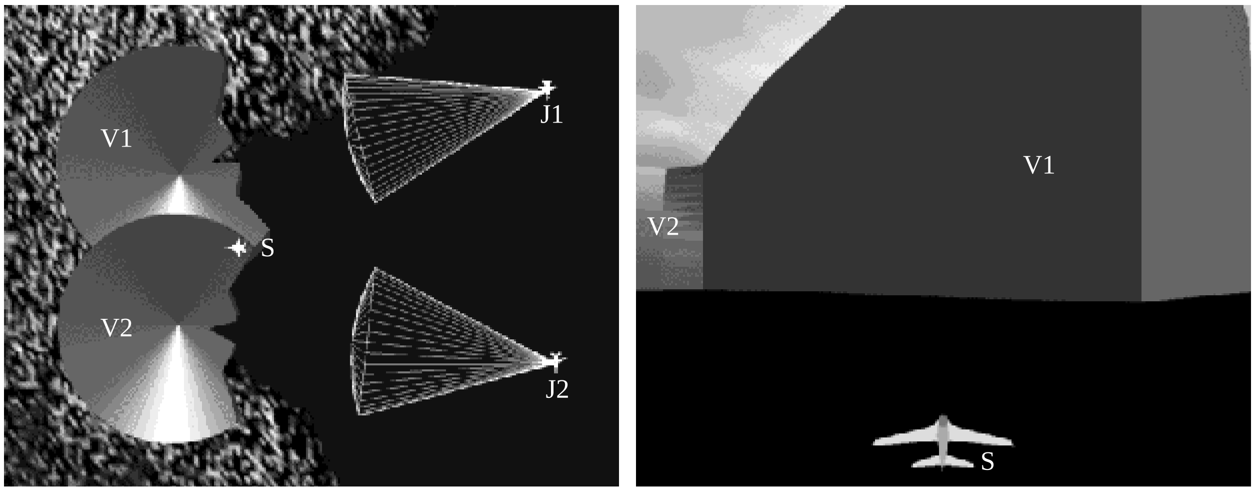

Figure 5: [Left] An example design configuration with the jammers (J1 and J2) too far from the SAM sites. The jamming is ineffective and does not open a sufficient corridor for the strike aircraft (S) between the detection volumes (V1 and V2). [Right] The immersive view of this configuration.

is discontinuous and interacts with the detection ranges in seemingly arbitrary ways. The fact that all aspects of the design are affected by altitude makes a VR visualization of the scenario particularly useful.

In actual practice, this problem would be significantly more complex. In addition to the parameters in the list above, the design parameters would actually include items such as missile firings (who fires what, when, and at whom), radar interactions (passive and active surveillance), decoy usage, and communications (friendly and hostile). The evaluation criteria would include the overall cost, a more robust measure of the probability of success, and the time needed for execution. The issue of scalability is generic to complex design systems and tasks. As the problem becomes unmanageable, the system may consider a subset of a design instead of the design in its entirety.

4.2. The Design Process

The design process begins with the user making a first “best guess” of the locations of the jamming aircraft and the strike trajectory. Figure 5 shows one possible initial design. The scenario must be simulated to compute the associated radar detection ranges of the missile sites and the land exposure of the jammers. A visualization of the results shows that the corridor is too narrow for the strike aircraft. The user believes that the cause is likely due to the positioning of the jammers and therefore suggests to the AI subsystem that jammer position has priority over strike path. In other words, the optimization function for this task could be chosen to be, for example:

$$X \times (\text{threat to J1}) + Y \times (\text{threat to J2}) + Z \times (P(\text{detection of S})) = N$$

where X,Y, and Z are weighting coefficients, P is the probability function, and N is the calculated objective evaluation measure. The user would adjust X,Y, and Z as to their relative importance to the solution.

After communicating suggestions to the AI subsystem (we will not speculate as to what the specific interaction techniques might be for this communication), one or more new candidate solutions are computed and available for simulation and visualization for the user. Based on the information given the AI subsystem in the previous iteration, each of these redesigns is "better" than the previous design (See the next section for a discussion of the AI processes that compute these redesigns). After several iterations, a strike plan is accepted (See figure 6). In this solution, the jammers are placed at positions where an acceptable strike corridor has been opened without subjecting the jamming aircraft to excessive land-based threats.

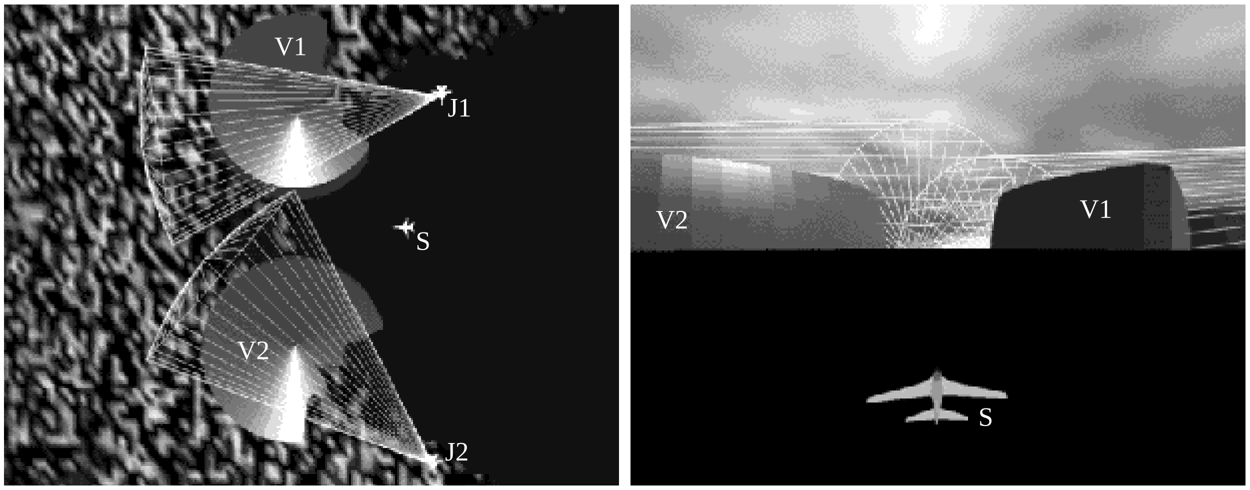

Figure 6: [Left] An acceptable design configuration with the jammers (J1 and J2) sufficiently far from the SAM sites and an appropriate corridor open between detection volumes (V1 and V2) for the strike aircraft (S). [Right] The immersive view of this configuration.

Current techniques would require that the user randomly move the jammers and re-execute the simulation to observe the results. The jammers could be placed at any elevation anywhere in the environment, including on the opposite side of the missile sites. The user will not usually know specifically what configurations will constitute a "better" design. This is the role of the proposed AI subsystem. In the case of our example, the AI subsystem will not produce candidate designs where the objective function is lower than any previous solutions. However, there is no guarantee that the system will produce a global maximization of the objective function. There may be local maximums based on the relative positioning of the jammers to each other that would be considered while a global maximum which requires placing a jammer in a radically different position, would not be considered. In general, with the current state of the art, only local maximization is practical for mid- to large-scale optimization tasks (tens of variables on up). Thus, the system produces the best solution "close" to the initial guess of the user, rather than the best possible overall.

4.3. A Solution

The solution to this problem involves specifying a curve (the strike force trajectory to and from the target) and two points (the jammer locations). It is not possible to optimize curves directly on a computer; they must be parametrized first. Taking the initial trajectory input by the user represented, say, as a spline curve, a neural network can be used to parametrize deviations from the user's first guess. Thus the user's guess plus the output of the neural network (where the input to the network is, for example, the distance along the user's curve) is the trajectory which the system will recommend. RBF (radial basis function) networks might be a good choice due to their local nature [7]. The number of RBF units included in the network, analogous to the number of knot points in a spline, will determine how flexible the trajectory is. The number can be set to a default and then decreased or increased as the user requests more or less smoothness. For this application, the basis functions could be centered at regular intervals. Alternatively, more functions (and thus more ability to adjust the trajectory) could be located according to a heuristic rule, for example more functions where the terrain is more hilly, and thus where the strike force might be expected to do a lot of turning to take advantage of the terrain. Figure 7 indicates the relationships between the initial trajectory guessed by the user, the one suggested by the AI subsystem, and the neural network used for modeling trajectory perturbations.

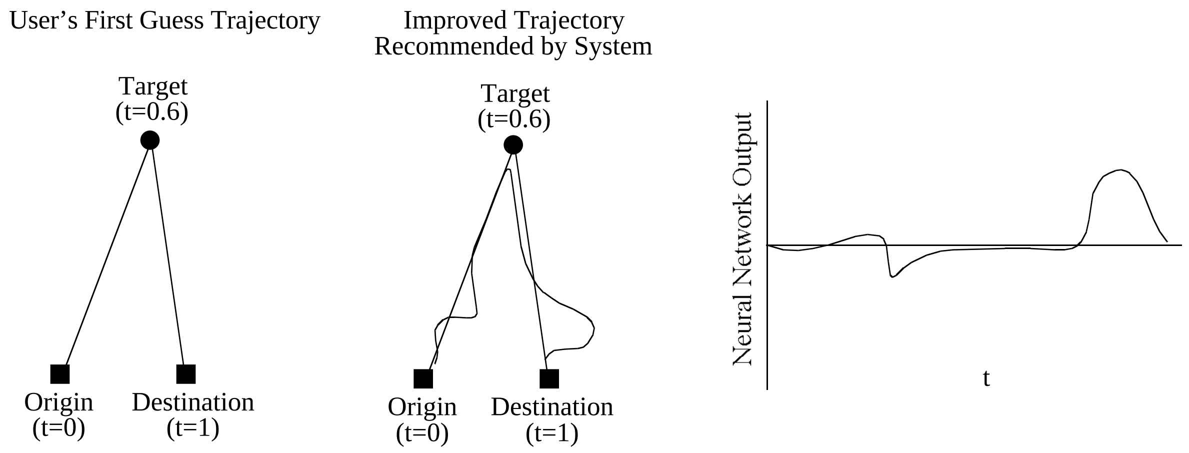

Figure 7: A neural network could be used to model perturbations to the initial trajectory suggested by the user.

The problem of improving the user's specified trajectory has now been reduced to a 6+N dimensional optimization problem, where the 6 accounts for the two jammer locations and N is the number of free parameters in the neural network. If the performance criteria are a differentiable function of these parameters, a function that calculates the derivative can be written by the user. Easier yet, the function might be automatically produced using a code differentiator such as ADOL-C [6]. Then any gradient-based optimization method can be used (gradient descent if high dimensional, and conjugate gradient or a Newton-like method if low dimensional). When derivatives either do not exist or can not be made available, sensitivity-style optimization, relying on "small" perturbations to the parameters is always an option. Heuristics may be used to speed this optimization. A possible example in this case might be to train a separate neural network to approximate some part of the simulator, for example to learn the dangerous region around each SAM installation as a function of jammer distance from the SAM site. By bypassing computationally expensive portions of the simulator in this manner, optimization speed can often be greatly improved at a modest cost in accuracy.

5. CONCLUSIONS

By applying the strengths of both VR and AI to engineering design problems, a system can be developed that may overcome some of the current limitation of such design systems. The system we have described here uses VR as a visualization tool for analysis and as a "natural" and intuitive way to view a design space and make suggestions as to its improvement. AI is then used to guide the designer's "search" for a correct, or at least acceptable, design. The system improves overall performance and shortens design cycle time by closing the loop between design tools and analysis tools; conceptually to the designer, they are one and the same. Eventually, when computational and graphics hardware permits, we believe that designs of highly complex systems can and will be developed on systems such as this in a relatively short amount of time as compared to current techniques.

ACKNOWLEDGEMENTS

We would like to thank the staff of ENEWS/NRL for their assistance in this work; especially Tobin Hill for his role as our domain expert and scenario builder. The images included were produced by the Interactive Scenario Builder developed by Tobin Hill and Brian Solan of ENEWS/NRL.

REFERENCES

[1] Bergman, L.D., *et al.*, *VIEW - An Exploratory Molecular Visualization System with User-Definable Interaction Sequences.* Computer Graphics, 1993. **27**(3): p. 117-126.

[2] Brooks, F.P. *Walkthrough - A Dynamic Graphics System for Simulating Virtual Buildings*. in *Workshop on Interactive 3D Graphics*. 1986. Chapel Hill, NC: ACM Press.

[3] Bryson, S. and C. Levit, *The Virtual Windtunnel: An Environment for the Exploration of Three-Dimensional Unsteady Flows*, 1991, National Aeronautics and Space Administration, Ames Research Center.

[4] Cruz-Neira, C. and P. Bash. *Integrating High Performance Computing and Communications with Virtual Reality for Interactive Molecular Modeling: The VIBE System*. in *Simulation and Interaction in Virtual Environments*. 1995. Iowa City, IA: ACM Press.

[5] Cruz-Neira, C., *et al. Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment*. in *Symposium on Research Frontiers in Virtual Reality*. 1993. San Jose, CA: IEEE Computer Society Press.

[6] Griewank, A., D. Juedes, and J. Utke. *ADOL-C, A Package for the Automatic Differentiation of Algorithms Written in C/C++*. To appear in ACM Transactions on Mathematical Software. See also the web site <http://www.math.tu-dresden.de/wir/project/wwwadolc/wwwadolc.html>.

[7] Hertz, J., A. Krogh, and R. Palmer. *Introduction to the Theory of Neural Computation*. 1991. Addison-Wesley Publishing Company.

[8] Narendra, K.S. and K. Parthasarathy. *Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks*. IEEE Transactions on Neural Networks, Vol. 2, No. 2, pp. 252-262. March, 1991.

[9] Rao, R.B. and S.C-Y. Lu. *Building Models to Support Synthesis in Early Stage Product Design*. AAAI-93, the Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 277-182. 1993.

[10]White, H. with A.R. Gallant, K. Hornik, M. Stinchcombe, and J. Wooldridge. *Artificial Neural Networks: Approximation and Learning Theory*. 1992. Blackwell Publishers.